ON THE COST OF APPROXIMATING AND RECOGNIZING A NOISE PERTURBED STRAIGHT LINE OR A QUADRATIC CURVE SEGMENT IN THE PLANE

by

David B. Cooper and Nese Yalabik

Division of Engineering
Brown University
Providence, Rhode Island 02912

March 1975

Abstract

Approximation of noisy data in the plane by straight lines or elliptic or single-branch hyperbolic curve segments arises in pattern recognition, data compaction, and other problems. A number of questions concerning the efficient search for and approximation of data by such curves are examined. Recursive least-squares linear curve-fitting is used, and ellipses and hyperbolas are parameterized as quadratic functions in x and y. The error minimized by the algorithm is interpreted. cpu times for estimating parameters for fitting straight lines and quadratic curves are determined and compared. cpu time for data search is also determined for the case of straight line fitting. Quadratic curve fitting is shown to require about six times as much cpu time as does straight line fitting. Curves relating cpu time and fitting error are determined for straight line fitting. Lastly, results are derived on early sequential determination of whether or not the underlying curve is a straight line.

(NASA-CR-142906) ON THE COST OF APPROXIMATING AND RECOGNIZING A NOISE PERTURBED STRAIGHT LINE OR A QUADRATIC CURVE SEGMENT IN THE PLANE (Brown Univ.) 49 p HC $3.75

N75-25642

Unclas
G3/63 24159

Introduction

One of the fundamental approaches to pattern recognition in general and to recognition of line drawings in particular is a statistical theoretic approach. Though other, usually heuristic, approaches are also taken, if one is interested in classification with controlled probability of error or with minimum computation cost, then a probabilistic approach is necessary. The question of minimizing cpu time for recognition, is of course, of great importance if one wishes to treat the recognition of highly complicated patterns and is a subject of growing interest [1-5]. In this paper we focus on a subproblem of the more general problem of optimally recognizing complicated line drawings, namely, the problem of recognizing underlying straight lines and quadratic arcs in line drawings. Before getting down to the problem at hand, we briefly comment on an approach to recognition of complicated line pictures as this will provide motivation for some of the topics we examine. However, it should be emphasized that our interest in approximating data by straight lines and quadratic curves is for many other applications as well, e.g., data compression in picture boundary representation, contour line representation in maps, etc. The methods we discuss are geared to handling very noisy data.

We assume a model for data generation in which there is an underlying picture, in two dimensions, consisting of straight lines and curves and a datum is a noisy perturbation of a point on one of the lines. If, e.g., the class of pictures under consideration is a class of handprinted capital letters, an appropriate perturbation model might be a second order stochastic process with the perturbation perpendicular to the underlying line and a correlation interval roughly the length of the underlying line. We assume the two-dimensional field of the picture is divided into cells, i.e., quantized in the x and y directions, and that the picture gray level is hard quantized. Hence, the data available to

the computer is a rectangular array of 1's and 0's -- a 1 if an appreciable portion of a cell is filled by the picture. A probabilistic description of the data is then given by a distribution for the underlying straight lines and curves and a distribution for the noisy perturbations of this picture. It is then in theory possible to construct a Bayes decision rule (minimum probability of misclassification decision rule) for classifying the data as representing a specific member of a finite number of families of pictures. This or the maximum a posteriori probability decision function will be of interest to us in the problems we treat. The maximum a posteriori probability decision function is easier to use and is essentially equivalent in practice since radical quantization of some of the variables is necessary in practice to sufficiently simplify the problem probability structure in order to use either method.

As an example of a number of considerations, suppose hand-printed capital letters are the pattern categories of interest. What are the likelihood functions involved in letter recognition? Let (x_i,y_i) be the ith data point and $w = (x_1,y_1,x_2,y_2,\ldots,x_n,y_n)^T$ be the observed data vector. There is a prior distribution for the occurrence of the 26 capital letters. The maximum a posteriori decision rule is "choose the letter for which the joint likelihood of the letter and w is at least as large as the joint likelihoods of each of the other letters and w". The joint likelihood of H and w might be specified as follows. Let P(H) denote the a priori probability of the occurrence of H. We assume the data, when H is true, is a noisy perturbation of three straight lines. Since each line can be specified by 2 endpoints -- 4 parameters -- the three lines can be specified by a parameter vector α of 12 components. Let $p_H(\alpha)$ denote the probability density function for the α vector specifying possible underlying H's. Let $f(w|\alpha,H)$ be the likelihood of the data given H and α. Then denoting the likeli-

hood of α and w given H by $p(\alpha,w|H)$, we define the joint maximum likelihood of H and w to be

$$\max_{\alpha} \; P(H)p(\alpha,w|H)$$

or

$$\max_{\alpha} \; P(H)p(\alpha|H)f(w|\alpha,H) \; . \qquad (1)$$

If we assume that the noisy perturbations for a line are independent of those for the other lines, the perturbation processes are the same for all lines, then (1) can be rewritten

$$\max_{\alpha} \; P(H)p(\alpha|H)f(w|\alpha_1)f(w|\alpha_2)f(w|\alpha_3) \; . \qquad (2)$$

Here, $\alpha_1,\alpha_2,\alpha_3$ denote the parameter vectors for lines 1, 2, and 3 and we assume that for each i, the components (x_i,y_i) of w appear in only one of the three densities $f(w|\alpha_1)$, $f(w|\alpha_2)$, and $f(w|\alpha_3)$, specifically, the density which results in (2) being a maximum. If the $f(w|\alpha_j)$ are Gaussian, and if, in addition, the data points are assumed to be <u>independent</u> perturbations of the underlying lines, then $f(w|\alpha_j)$ is simply $(2\pi\sigma^2)^{-2n_j}\exp[-(1/2\sigma^2)q(w,\alpha_j)]$ where n_j is the number of data points appearing in $f(w|\alpha_j)$, and $q(w,\alpha_j)$ is the sum of the squares of the distances of these data points to the line determined by the parameters α_j. Note that if $p(\alpha|H)$ were not present, (2) would be maximized by choosing the α_j's such that the lines they specify are the least squares fits to the associated data points in w. The presence of $p(\alpha|H)$ modifies the best α somewhat and also plays a role in determining the association of the data points in w and the 3 distributions $f(w|\alpha_j)$, $j = 1,2,3$.

The problem with implementing (2) for all 26 letters is that the computation can become costly. Three important considerations arise when one attempts to reduce the amount of computations. First, how many data points should be used?

If the data points are assumed to be independent perturbations of underlying lines (or curves), the classification error will be a decreasing function of n, the number of data points. If the data points are not independent perturbations, classification error decreases and then levels off as n increases. Since computation cost increases with increasing n, in general it will be desirable to use only a subset of the available data. Second, computing (2) for all 26 letters is very costly. Third, determining the distribution $f(\cdot|\alpha_j)$ with which a data point should be associated can be costly. This is related to the so called segmentation problem but in some aspects is more complicated if the recognizer processes only a subset of the data and hence does not exploit sequential continuity when it exists. The complexity of these problems has led researchers to turn to sequential recognition. Here it has been realized that by fitting lines and curves sequentially, it is possible to rule out various letters as being highly unlikely along the way, thus avoiding the second high cost referred to. Sequential recognition results in other benefits as well.

We examine the preceding three considerations more carefully within the context of sequential recognition. We assume that recognition starts with the 0,1 quantized data matrix and the goal then is low computational cost recognition. What is usually considered as preprocessing, e.g., smoothing and determination of data points constituting a curve through use of local connectivity and curvature, normalization, etc, usually involve processing all the data and are costly operations and ones we believe should be included in the total recognition cost and treated within the recognition cost minimization framework. The approach here, then, is sequential determination of the maximum likelihood function for the letter classes and the data. Only a subset of the available data is processed in general. Under the assumption that the $f(w|\alpha_j)$ are Gaussian, the choice of

letter and parameters for which (2) is a maximum is equivalent to the problem of optimally sequentially fitting straight lines and parameterized arcs to subsets of the data. (i) Assume the recognition process is at a stage where a straight line or parameterized curve is to be fit to data entering a specific local region. The appropriate data subset is not yet known and must be found during the fitting process. Hence, influenced somewhat by the results of the fit up to the present stage, the system searches for data in the specified local region, gets a preliminary fit of a straight line or other parametrized curve to this data and then uses the fitted curve to estimate the location of the next data point to be used. Starting from the estimated location of the new data point, the nearest data point is found through a search, the parameters for a better fitting curve to the augmented data are found, and a new data point then looked for. Hence this provides a way of exploiting prior knowledge of pattern structure for intelligently searching for the data subset associated with an underlying parameterized curve. If the data perturbations are large, the cpu time consumed in data search when fitting straight lines to data can be comparable to the cpu time in computing parameter values which result in the least squares straight line fit to the data. When the parameterized curve is a quadratic curve, the cost of the data search is less than the cost of computing appropriate parameter values. This problem of intelligent data search arises in a variety of contexts, e.g., in ballistic missle decoy tracking [6]. (ii) There is a least expected cost order in which to search for the underlying parameterized curves. In otherwords, optimal sequential parametrized curve fitting can be directed through a <u>least expected cost</u> decision tree. See [2] as an example of the type of minimum cost decision tree approach one can consider here. The reason for this dependency of recognition cpu time on order of estimation of the underlying curves is twofold. First, the a priori letter probabilities and hence probabilities of occurrence of various

combinations of underlying parameterized curves vary. Second, the four major operations in curve fitting are: checking to see whether there is an underlying curve in the vicinity of a specific region of the space; fitting a straight line to data; checking to see which of a line and a quadratic curve best fits the data; fitting a higher order parameterized curve to data. These operations require different amounts of cpu time with the first requiring the least cpu time and the last requiring many times that amount. Hence, e.g., if recognition can be accomplished by showing that the joint likelihood of a letter and the data w is unacceptably low for all 25 of the 26 letters, the recognition decision will be the remaining letter without actually computing the likelihood of this letter and its data. Thus, an optimum curve fitting sequence may avoid fitting costly quadratic or higher order curves and rely more for recognition on fitting straight lines, where they exist, checking on whether or not curves are present in specific regions, and fitting quadratic curves only when absolutely necessary. The optimum decision tree depends on the various probabilities involved and on the costs of performing various operations. (iii) Finally, how well does one estimate each underlying curve? Equivalently, how many data points do we take for estimating each underlying curve? We have some choice here since determining the best possible fit to the underlying curves is usually not necessary for acceptable recognition error rate. On the other hand, it is usually true that the better the curve fit up to a stage, say the ith, the less uncertainty and hence less costly will be the fit at the following stage; also, the higher will be the probability of correct underlying curve determination at the ith stage and hence the lower the probability for incorrect fitting and hence costly backtracking at the next or other future stages. Consequently, it behooves us to understand the relationship between goodness of curve fit and associated computational cost in order to determine how good a fit to compute at each stage. This problem of the

relation among computation and resulting uncertainty at a given stage and the required computation at the next stage is important and arises in a variety of ways in a pattern recognition problem. See [5] for interesting treatment of the tradeoffs in the two stages of letter recognition and then word recognition.

We do not study the preceding three points within the context of a complex abstract pattern recognition problem since the application context has a bearing on how the three points are best handled. However, we do provide results on the three points which are of use when considering a complete pattern recognition problem. Specifically, we discuss efficient algorithms for sequentially fitting straight lines and quadratic curves and for searching out the appropriate data. For straight lines and quadratic curves we estimate the cpu time for computing best parameter values, and for lines we also estimate cpu time for the data search. Thus, the relative data search and parameter estimation costs can be compared and the relative estimation costs for straight lines and quadratic curves can be compared. For straight lines, we arrive at graphs for fit error as a function of computation time. Such design graphs can be prepared for quadratic curves as well. Finally, we discuss a variety of procedures for early decision on whether an underlying straight line or nonlinear function best fits the data.

Models

We briefly describe the two curve fitting models of interest in this paper. Both are linear regression models. (i) First, we are interested in the least squares fit of a straight line to n data points. The most reasonable error measure is undoubtedly the perpendicular distance from a data point to the line. However, results are simpler to obtain if we parametrize y as a function of x or x as a function of y depending on whether the slope of the fitted line, at that time, has magnitude less than or greater than 1. These parameterizations are $y = c_o + c_1 x$ and $x = c_o + c_1 y$, respectively. Then for the first parameterization, the error for a given data point (x_i, y_i) is the magnitude of the difference in y_i and the height of the approximating line at x_i; the error measure for the second parameterization is the obvious analog. In practice, one would start off with the most reasonable parameterization based on available information, and then retain this parameterization or change it depending on the data being processed. Since the sum of the squares of the errors in the best fit of this type is related by at most a factor of 2 to the corresponding statistic when the perpendicular error from data point to line is used, it can be seen that, visually or by any reasonable criteria, the resulting line should be perfectly satisfactory.
(ii) The second model of interest is the fitting of data by a quadratic curve. A portion of an ellipse is most useful for our purposes. The parameterization $x(t) = x_o + a \cos t$, $y(t) = y_o + b \cos t + c \sin t$ has many desirable features and variations have been used, e.g., [7]. This parameterization is probably the most convenient to use if successive values of t are chosen first and then appropriate data points are searched for. That is, the approach is most useful if the search for an ith data point is conducted by calculating best estimates for x_o, y_o, a, b, and c based on the first i-1 data points used and then choosing t_i and searching for the data point closest, in the perpendicular direction, to the

estimated ellipse at t_i. However, if one is faced with the problem of fitting an ellipse to n specific data points, this parameterization is inconvenient since the appropriate value of t to be associated with each data point is not known and the fitting operation then involves some nonlinear programming. An alternative procedure is to ask for the coefficients for which the sum of the perpendicular distances squared from n data point to the curve

$$c_1x^2 + c_2xy + c_3y^2 + c_4x + c_5y - 1 = 0 \qquad (3)$$

is a minimum. Again, the solution can be found but the required computation is considerably greater than if an alternative error measure, which we now describe, is used.[1] Let c denote the vector $(c_1, c_2, \ldots, c_5)^T$ and z denote the vector $(x^2, xy, y^2, x, y)^T$. Let z^i denote the z vector for the point (x_i, y_i). Then we seek the c vector for which

$$\sum_{i=1}^{n} (c^T z^i - 1)^2 \qquad (4)$$

is a minimum. This formulation can of course be used for a straight line, for curves of higher order or for other parameterizations as well. What is the curve resulting from miminization of (4)? The curve can be any one of an ellipse, a hyperbola, or a parabola. The chance of the last of these occurring is essentially zero. We would like to obtain an ellipse and in the section on experimental results we discuss ways of coaxing the algorithm to generate an ellipse or at least a <u>useful</u> hyperbola.

For c fixed, c^Tz is a function of (x,y), and, hence, $w-c^Tz = 0$ is a surface in Euclidean w,x,y-space. The error measure (4) is the sum of the squares of the distances of the points $w_i \equiv c^Tz^i$, on the surface $w-c^Tz = 0$, to the plane parallel to and one unit above the x,y plane through the origin. See Fig. 1.

The intersection of the surface $w-c^Tz = 0$ with a plane parallel to the x,y plane through the origin is a quadratic curve $c^Tz + d = 0$ or is empty -- depending on d. Suppose, for example, that (3) is an ellipse. Then as d varies, the curves $c^Tz + d = 0$ consitute a family of ellipses with common center, same major axes and minor axes, and such that the ratio of the distances of the curve from its center along the major and minor axes is the same for all d. An alternative and more useful interpretation of (4) is the following. Consider the ellipse and data $v_d \equiv (x_d,y_d)^T$ in Fig. 2. Let $v_o \equiv (x_o,y_o)^T$ denote the ellipse center and $v_e \equiv (x_e,y_e)^T$ denote the intersection of the ellipse with the straight line from v_o to v_d. c^Tz-1 can be rewritten as

$$(v-v_o)^T C(v-v_o) - v_o^TCv_o - 1 \qquad (5)$$

where

$$C \equiv \begin{bmatrix} c_1 & \frac{1}{2}c_2 \\ \frac{1}{2}c_2 & c_3 \end{bmatrix} \quad \text{and} \quad v_o = -(1/2)C^{-1}\begin{bmatrix} c_4 \\ c_5 \end{bmatrix}.$$

Upon defining u_e to be v_e-v_o and δ to be a scalar such that $v_d-v_o = u_e + \Delta u_e$, (5) becomes $(v_e-v_o)^TC(v_e-v_o) - v_o^TCv_o-1 + \delta u_e^TC\delta u_e + 2\delta u_e^TCu_e$, or $\delta(\delta+2)u_e^TCu_e$. Finally, since $u_e^TCu_e-v_o^TCv_o-1 = 0$, (5) can be written as

$$\delta(\delta+2)\rho \qquad (6)$$

where ρ is the constant $1+v_o^TCv_o$. Note that $|\delta|$ is the ratio $||v_d-v_e||/||u_e||$ and, hence, is the ratio of the distance of the data point to the ellipse, along the line from the data point to the ellipse center, divided by the distance from the ellipse to the ellipse center along this line. See Fig. 2. Since $||u_e||$ is a maximum in the direction of the ellipse major axis and a minimum in the direction of the ellipse minor axis, it can be seen that if two data points are equidistant

from the ellipse but one lies along the ellipse major axis and the other along the ellipse minor axis, then the contribution to (4) of the data point lying along the ellipse minor axis will be greater, assuming $|\delta| << 1$. Also, for $|\delta|$ fixed, (6) has larger magnitude for $\delta > 0$ than for $\delta < 0$. The difference in magnitudes is negligible for $|\delta| << 1$.

Recursive Estimation

Estimation of the parameters of the models $y = c_0+c_1x$ and $c_1x^2 + c_2xy + c_3y^2 + c_4x + c_5y - 1 = 0$ is efficiently realized through use of recursive least squares estimation. Specifically, letting

$$A_n = [z^1 \mid z^2 \mid \ldots \mid z^n]^T , \quad \text{and } b = (y^1,y^2,\ldots,y^n)^T$$

or $b = (1,1,\ldots,1)^T$, n 1's, depending on whether we are dealing with the first or the second model, respectively, the problem of determining the optimum c is that of determining the c which minimizes $||A_nc - b||^2$. The solution from [8] or [9] is

$$c = A_n^+b \tag{7}$$

where A_n^+ denotes the pseudo inverse $(A_n^TA_n)^{-1}A_n^T$. However, from [10] we see that $(A_n^TA_n)^{-1}$ can be computed recursively. Let B_n denote $A_n^TA_n$. Then

$$B_{n+1}^{-1} \equiv \left(\sum_{i=1}^{n+1} z^iz^{iT}\right)^{-1} = B_n^{-1} - \frac{B_n^{-1}z^{n+1}z^{n+1T}B_n^{-1}}{1+z^{n+1T}B_n^{-1}z^{n+1}} . \tag{8}$$

Since $Z^n \equiv A_n^Tb^n = \sum_{i=1}^{n} z^ib_i$, we see that the optimum c^{n+1} is given by

$$c^{n+1} = B_{n+1}^{-1}Z^{n+1} = \left[B_n^{-1} - \frac{B_n^{-1}z^{n+1}z^{n+1T}B_n^{-1}}{1+z^{n+1T}B_n^{-1}z^{n+1}}\right]\left[Z^n+b^{n+1}z^{n+1}\right] . \tag{9}$$

Hence, B_n^{-1} and Z^n are efficiently computed recursively, and c^{n+1} is computed in terms of these functions and the latest data, i.e., b^{n+1} and z^{n+1}.

The error $\mathcal{E}_n \equiv ||A_n c^n - b^n||^2$ can also be computed efficiently. Since

$$\mathcal{E}_{n+1} = \sum_{k=1}^{n+1} b_k^2 - 2 \left(\sum_{k=1}^{n+1} b_k z^k\right)^T c^{n+1} + \sum_{k=1}^{n+1} (z^{k^T} c^{n+1})^2 ,$$

it is trivial to show that

$$\mathcal{E}_{n+1} = ||b^{n+1}||^2 - Z^{n+1^T} B_{n+1}^{-1} Z^{n+1} = ||b^{n+1}||^2 - Z^{n+1^T} c^{n+1} \qquad (10)$$

where $||b^{n+1}||^2$, Z^{n+1}, B_{n+1}^{-1} are all trivially recursively updatable so that $\mathcal{E}_{n+1}$ is computed in terms of $||b^n||^2$, Z^n, B_n^{-1}, b_{n+1}, and z^{n+1}. Note that almost all of the computational effort for c^{n+1} and $\mathcal{E}_{n+1}$ is accounted for in the computation of B_{n+1}^{-1} and then $B_{n+1}^{-1} Z^{n+1}$. Since this computation is needed for the determination of both c^{n+1} and $\mathcal{E}_{n+1}$, it follows that once one computes one of these estimates, the other is computed cheaply.

Computation Costs

Since recursive estimation schemes are under consideration, we estimate the CPU time for one such recursion. We deal with four costs, these being:

(i) cost of computing c^{n+1}

(ii) additional cost of computing $\mathcal{E}_{n+1}$

(iii) cost of predicting best initial search point for new data point search

(iv) new data point search cost given first search point.

All costs are estimated by determining the numbers of various types and associated CPU realization times for arithmetic, Boolean and memory management operations. CPU realization times are for assembly language programs and are taken from [11]. The instruction list and associated realization times are given in Table 1.

Table 2 contains a list of the numbers of the various instructions determining the four aforementioned costs and the resulting costs. A short explanation of the details of these calculations is the following. The inversion of B_{n+1}^{-1} is carried out by first computing $B_n^{-1}z^{n+1}$. This vector is then used in the computation of (8). Since $z^{n+1} = (1,x_{n+1})^T$ when fitting a straight line, only two multiplications are required in computing $B_n^{-1}z^{n+1}$ in this case. Since $B_n^{-1}z^{n+1}\ z^{n+1^T}B_n^{-1}$ is symmetric, only the diagonal and half the off-diagonal entries must be calculated. Finally, it is not necessary to divide each component of $B_n^{-1}z^{n+1}z^{n+1^T}B_n^{-1}$ by $1+z^{n+1^T}B_n^{-1}z^{n+1}$. Dividing the vector $B_n^{-1}z^{n+1}$ by $1+z^{n+1^T}B_n^{-1}z^{n+1}$ and then computing $[(1+z^{n+1^T}B_n^{-1}z^{n+1})^{-1}B_n^{-1}z^{n+1}]z^{n+1^T}B_n^{-1}$ is more efficient.

Now consider costs (iii) and (iv). For model 1, a choice is made for x_{n+1} and then a search made for a data point having this x_{n+1}. If the data is complex consisting of more than one noisy line and arc, the data point of interest is that closest to the line determined by c^n. If there is only one line and no arcs presented, the only consideration is a least cost search for the data point and such a search, for data generated by Gaussian perturbations of a straight line, involves a good initial estimate $\hat{y}_{n+1}$ followed by successive examination of cells above and below those last explored. The necessary estimate in the first case, which is also the minimum mean squared error estimate to be used in the second case, is

$$\hat{y}_{n+1} = z^{n+1^T}c^n \ . \tag{11}$$

In practice, x_{n+1} would probably be chosen such that the differences $x_{n+1} - x_n$ are equal and this constant chosen at the start of the search. For model 2, the preceding considerations also apply. The initial choice we consider in the search for a z^{n+1} is a point $\hat{z}^{n+1}$ on the curve

$$\hat{z}^{n+1^T}c^n = 1 \ . \tag{12}$$

Such a point might be found by choosing an x_{n+1} or a y_{n+1} and then solving (12) to determine the other and, hence, $\hat{z}^{n+1}$. The search for z^{n+1} is then probably most cheaply carried out by determining and then searching in the direction perpendicular to the curve at $\hat{z}^{n+1}$. Alternative less accurate choices for $\hat{y}^{n+1}$ and $\hat{z}^{n+1}$ might be considered, although the saving in costs for computing these estimates is small compared with the total computational cost of one recursion, and, hence, not a significant consideration. For example, the cost of calculating $\hat{y}_{n+1} = z^{n+1^T} c^n$ is essentially the cost of <u>one</u> multiplication.

For the 1st model, the search for (x_{n+1},y_{n+1}) is the sequence $(x_{n+1},\hat{y}_{n+1}+\Delta)$, $(x_{n+1},\hat{y}_{n+1}-\Delta)$, $(x_{n+1},\hat{y}_{n+1}+2\Delta)$, etc., where Δ is the quantization interval. Under the assumption that $y_{n+1}-z^{n+1^T}c$ is a Gaussian r.v., with $c = (c_o,c_1)^T$ the true parameter vector, the specified search is of lowest cost. Assuming the variance σ^2 of $y_{n+1}-(c_o+c_1x_{n+1})$ given x_{n+1} is much greated than Δ^2, an estimate of the expectation of the number of iterations to finding y_{n+1} is obtained as follows. y_{n+1} and $\hat{y}_{n+1} = z^{n+1^T} c^n$ are independent Gaussian random variables with common expectation $z^{n+1^T}c$. The variance of y_{n+1} is of course, σ^2, and that of $\hat{y}_{n+1}$ is var c_o^n + $(x_{n+1})^2$var c_1^n [12]. Note that by c_o^n and c_1^n we mean the first and second components of c^n. Hence,

var $\hat{y}_{n+1} = \sigma^2[n^{-1} + (x_{n+1} - \bar{x}_n)^2 / \sum_{i=1}^{n} (x_i-\bar{x}_n)^2]$, and $\sigma^2_{n+1} \equiv$ var$(y_{n+1}-\hat{y}_{n+1})$

= var y_{n+1} + var $\hat{y}_{n+1} = \sigma^2[1+n^{-1} + (x_{n+1} - \bar{x}_n)^2 / \sum_{i=1}^{n} (x_i-\bar{x}_n)^2]$. ($\bar{x}_n$ denotes $n^{-1}\sum_{i=1}^{n} x_i$.) The expected number of iterations is

$$P\{y_{n+1}-\hat{y}_{n+1} = 1\} + 2\cdot P\{y_{n+1}-\hat{y}_{n+1} = -1\} + 3\cdot P\{y_{n+1}-\hat{y}_{n+1} = 2\} + \ldots$$

which is roughly

$$2\sum_{i=0}^{\infty} 2i\Delta(2\pi\sigma^2_{n+1})^{-1}\exp[-(1/2\sigma^2_{n+1})i^2\Delta^2] \approx 4(2\pi)^{-1/2}\sigma_{n+1}/\Delta \qquad (13)$$

for $\sigma/\Delta >> 1$. Note that $\sigma/\Delta >> 1 \Rightarrow \sigma_{n+1}/\Delta >> 1$ which then implies that the probability of the events $y_{n+1} = \hat{y}_{n+1} + i\Delta$ and $y_{n+1} = \hat{y}_{n+1} - i\Delta$ are each approximately $\Delta(2\pi\sigma^2_{n+1})^{-1/2}\exp[-(1/2\sigma^2_{n+1})i^2\Delta^2]$ and the sum in (13) can be approximated by an integration. When $x_i - x_{i-1}$ is a constant for all $i \leq n$,

$$\sigma^2_{n+1} = \sigma^2(n^3 + 12n^2 + 20n + 4)/(n^3 + 8n^2) . \qquad (14)$$

Then, the expected number of iterations is roughly

$$4(2\pi)^{-1/2}(\sigma/\Delta)[(n^3 + 12n^2 + 20n + 4)/(n^3 + 8n^2)]^{1/2} . \qquad (15)$$

As $n \to \infty$, (15) converges to $4(2\pi)^{-1/2}(\sigma/\Delta)$ and is indeed close to this value for $n \geq 5$.

From Table 2 note that the sum of costs (i), (ii), and (iii) is roughly six times as great for a quadratic arc as for a straight line.

Variability In Straight Line Approximation

The variability in the fitted line $y = z^T c^n$ depends on the number and specific choice of the x_i's. In order to make an intelligent choice for the x_i, a measure of fitted line variability is needed. Under the assumption that the y_i are perturbations of some line $c_o + c_1 x_i$, a useful measure of fitted line variability is

$$\{E[(z^{1^T}c^n - z^{1^T}c)^2 + (z^{n^T}c^n - z^{n^T}c)^2]\}^{1/2} \qquad (16)$$

which is the square root of the sum of the variances of the y components of the fitted line endpoints. (The assumption here is that $|c_1| \leq 1$; otherwise, the variances in the x components of the fitted line endpoints are of interest.) The evaluation of (16) is like that of σ^2_{n+1} and is easily shown to be

$$\sigma\{2n^{-1} + [(x_1 - \bar{x}_n)^2 + (x_n - \bar{x}_n)^2]/\sum_{i=1}^{n}(x_i - \bar{x}_n)^2\}^{1/2}$$

which reduces to

$$\sigma[2n^{-1} + 2(x_n-\bar{x}_n)^2/\sum_{i=1}^{n}(x_i-\bar{x}_n)^2]^{1/2} , \tag{17}$$

and is

$$\sigma[2(4n^2 + 20n + 4)/(n^3 + 8n^2)]^{1/2} \tag{18}$$

when x_i-x_{i-1} is constant for all $i \leq n$. A more useful measure of variability is (16)/L or (14)/(x_n-x_1) where L is the length of the approximating line. Since, for the parameterization under consideration, the slope of the approximating line has magnitude less than 1, the two measures are essentially equal. Hence, for convenience, we shall use

$$[\sigma/(x_n-x_1)][2n^{-1} + 2(x_n-\bar{x}_n)^2/\sum_{i=1}^{n}(x_i-\bar{x}_n)^2]^{1/2} \tag{19}$$

or

$$[\sigma/(x_n-x_1)][2(4n^2 + 20n + 4)/(n^3 + 8n^2)]^{1/2} , \tag{20}$$

when the x_i-x_{i-1} are equal for all i, as our measure of straight line fitting variability or "error". The advantage of (19) over (17) is that division of (17) by x_n-x_1 appropriately normalizes the measure of endpoint variability. That is, as long as the standard deviation of the endpoint of the approximating straight line is much smaller than is the approximating line length, we have a useful approximation and are justified in characterizing the error or the fitting variability as being small.

Variability In Straight Line Approximation As A Function of Computation Time

Since most of the computation time in identifying and approximating, by a parameterized line, a straight line or an arc is spent in the approximating process, we examine the relationship between curve fitting error and computation time. We do this, however, for straight lines only, and furthermore, for x_i is such that x_i-x_{i-1} is a constant for all i. Consequently, for the error measure we use (20). Since

line fitting is realized recursively with one recursion for each data point used, the total computation time is the sum of the computation times for each recursion. There are three computation costs which are the same for each recursion, and a fourth which is a function of the number of recursions previously completed but which becomes essentially constant after four or five recursions. We denote the first 3 costs per recursion as t_c, t_e, and t_p with t_c the cost of computing the estimate of the parameter vector c, t_e the cost of computing the sum of the squares of the distances from data points used to the latest best line fit, and t_p the cost of computing the best first search point for the new data-point search. The fourth cost, $t_s(i)$, is the cost of searching for the i-th data point given the best starting point. As mentioned, $t_s(i)$ is essentially constant for $i \geq 5$. If n data points are used in the fitting process, the fitting cost is roughly

$$n(t_c + t_e + t_p) + \sum_{i=1}^{n} t_s(i) \tag{21}$$

which is approximately

$$n(t_c + t_e + t_p + t_s) . \tag{22}$$

Before evaluating (21) and (22), we make a few observations on the relationship of error to cost. Among the situations which can be encountered are two extremes: (i) $t_c + t_e + t_p \gg t_s$, and (ii) $t_c + t_e + t_p \ll t_s$. Suppose (i) is true. Then n is directly proportional to fitting cost. Since error varies roughly as a constant multiplied by $n^{-1/2}$, we see that error varies as $(\text{cost})^{-1/2}$ multiplied by a constant and this constant is uniquely determined by the parameter $\sigma/(x_n-x_1)$. Suppose (ii) is true. Then for $n \gg 5$, a reasonable approximation to fitting cost is nt_s. However, since t_s is directly proportional to σ, then again using the fact that error is roughly constant$\cdot n^{-1/2}$, we conclude that error is roughly constant$(\sigma^{3/2}(x_n-x_1)^{-1})$ $(\text{cost})^{-1/2}$. Hence, error then is a function of the parameter $\sigma^{3/2}/(x_n-x_1)$. A typical curve of error versus cost is that of Fig. 3.

Stopping Procedures

The process of fitting a line or a quadratic curve to data was treated in the preceding sections. Two questions remain. First, how should the computer make a decision as to which of a line or quadratic curve best fits a subset of the data, and how useful is this data subset to the recognition of the entire pattern? Second, how does the computer decide, during the sequential fitting process, when to terminate the fitting of a line or quadratic curve because the latest data being precessed is that for a new line or curve? Under the assumption that good line and quadratic curve fits are possible, i.e., parameter estimation error is small for estimates based on a reasonable number of data points, stopping decisions can be made at costs smaller than the curve fitting costs. In this case, attention need be given only to the first question. If a curve fit is carried out with large estimated variance because an insufficient number of data points is used or is available, then terminating the fitting of a curve could be rather costly and could require processing much of the data used in fitting the intersecting curve which comes next. We assume that the former case pertains. Thus, the problem to be considered is to decide as early as possible whether the data used in a curve fitting is that for a line or for a quadratic curve. How does previously collected data affect the decision? Roughly for cases (i) and (ii) treated in the following paragraphs, based on the previously collected data or on some function of it, we have a conditional joint distribution for whether the underlying curve is a line or a quadratic and for the associated unknown parameters. Denote this density $p(H,c|w)$ where H denotes the hypothesis and takes values H_o and H_1 for line and quadratic, respectively, and c is the parameter vector for the curve type for the associated hypothesis. w denotes previously accumulated or reduced data. Given w, the joint likelihood of H,c and the most recently acquired

data $y_1, y_2, \ldots, y_n$ to which we are interested in fitting a curve is

$$p(H,c|w)f(x_1, \ldots, x_n|H,c) \qquad (23)$$

where we are assuming the data as perturbations of the underlying curve is independent of w given H,c.

For reasons discussed in the introduction, the stopping rules we briefly examined are those involving maximum likelihood ratios. Also, in general we are apt to be interested in a multihypothesis problem involving a decision as to which of a number of straight lines or arcs best fits a data subset entering a local region. Such a decision will be made by comparing each of one or more discriminant functions with a threshold where a threshold will depend on the discriminant function being tested and may depend on the number of data points and the extent of the region over which they are taken. (See [13] for some comments on abstract multihypothesis sequential decision theory.) In practice, the thresholds would probably be determined partially through experimentation. Model (3) is most easily studied and an understanding of its stopping behavior should provide a rough understanding of the stopping behavior of (4). We consider three versions of the simple problem of deciding whether the data constitutes a noisy perturbation of a linear function of x or of a quadratic or perhaps less restricted function of x. The three versions represent three different assumptions of prior structural knowledge of the underlying curves in x. Prohlem (i). We assume the underlying curve is $y = c_o + c_1x$ under H_o and is $y = c_o + c_1x + c_2x^2$ with c_2 known under H_1. c_o and c_1 are different under H_o and H_1, are unknown and may enjoy large ranges of possible values. The assumption of c_2 known is made in order that we be able to devise a test to distinguish between linear functions of x and functions of x which have at least some minimum quadratic content of interest. If the true c_2 is in fact larger in magnitude than the c_2 for which the decision function is designed, then the decisions made will have lower error probability than that

designed for. Problem (ii). Two hypotheses as in (i), but c_2 is not considered known. Problem (iii). A linear function is assumed under H_o, and no assumption is made concerning the form of the underlying curve under H_1. Hence, the assumptions range from somewhat restrictive in (i) to very unrestrictive in (iii).

The decision procedure then for (i) or (ii) is

$$H_1 \text{ if } \frac{\max_{\beta} p(H_1,\beta|w)f(y_1,\ldots,y_n|H_1,\beta)}{\max_{\gamma} p(H_o,\gamma|w)f(y_1,\ldots,y_n|H_o,\gamma)} > 1 \qquad (24)$$

H_o otherwise.

Equivalently, the log of the left side of (24) can be compared with 0. In practice, $p(H,c|w)$ will be of use only if it is of very simple form. This most likely restricts it to being Gaussian in its dependence on c or a function which takes constant values over a few simple regions in H,c space. For example, a particularly simple function, Fig. 4, might treat all lines having leftmost ends close to vertical A and right most ends in the shaded region as equally likely. The function might have one or more such regions of fairly high probability and the remainder of the space have likelihood 0.

A convenient form for viewing (24) is the following. Under our white Gaussian noise perturbation assumption, $p(H_o,\gamma|w)f(y_1,\ldots,y_n|H_o,\gamma)$ is

$$p(H_o,\gamma|w)(2\pi\sigma^2)^{-n/2}\exp[-(1/2\sigma^2)\sum_{i=1}^{n}(y_i - \gamma_o - \gamma_1 x_i)^2] \qquad (25)$$

which can be reformed as

$$p(H_o,\gamma|w)\exp\{-(1/2\sigma^2)\sum_{i=1}^{n}[\hat{c}_o - \gamma_o + (\hat{c}_1 - \gamma_1)x_i]^2\} \times$$
$$(2\pi\sigma^2)^{-n/2}\exp[-(1/2\sigma^2)\sum_{i=1}^{n}(y_i - \hat{c}_o - \hat{c}_1 x_i)^2] , \qquad (26)$$

where $\hat{c}_o$ and $\hat{c}_1$ are the parameters for a least squares line fit to the data $y_1,\ldots,y_n$. Hence, the influence of w on the maximization of (26) with respect to γ appears in the first line of (26). The second line of (26) involves only the latest data subset and the least squares line fit to that data. If the value of γ at which (26) is a maximum is determined largely by the exponential in the first line of (26), then $p(H_o,\gamma|w)$ in the denominator of (24) can be placed before the maximization. If $p(H_1,\beta|w)$ can similarly be removed from the maximization in the numerator of (24), the effect of w is simply to multiply

$$\frac{\max_{\beta} f(y_1,\ldots,y_n|H_1,\beta)}{\max_{\gamma} f(y_1,\ldots,y_n|H_o,\gamma)} \tag{27}$$

by a constant. This situation is likely to occur when n is large. (This does not mean that w has not played an important role. It plays an important role in guiding the search for $y_1,\ldots,y_n$ and then in weighting (27).) Consequently, we examine some properties of decision making based on (27).

The x values x_1, x_2,..., at which data points are sought are chosen first, based on prior knowledge of the lengths of lines or curves likely to be encountered. In practice, equal increments x_i-x_{i-1} would probably be used and one can determine a best (in some sense) increment to use. We do not do that here. The object of this section is to provide some insight into the structure of good decision rules for deciding with a minimum number of data points whether the latest data subset is best fit by a line or a nonlinear arc.

Case (i): For this case we employ a Wald sequential probability ratio test (SPRT) [14] having the property that the decision for accepting H_o or H_1 is made with predetermined probabilities of incorrect decision under H_o and H_1, set by the designer, and this performance is achieved by processing the minimum expected number of data points. The parameter vector c^n giving the best linear

fit to the data has previously been found to be (9), i.e.,

$$c^n = B_n^{-1} z^n .$$

If the quadratic approximation is used with known coefficient c_2 for the second-power term, the least squares estimate for (c_o, c_1) is given by

$$c'^n = B_n^{-1} \sum_{i=1}^{n} z^i (y_i - c_2 x_i^2) \quad , \quad \text{or}$$

$$c'^n = c^n - B_n^{-1} \sum_{i=1}^{n} z^i c_2 x_i^2 \quad . \tag{28}$$

Then a sequential maximum likelihood ratio test consists of comparison of the log of the maximum likelihood ratio, (27), with a pair of stopping boundaries. The log of (27) is (29) multiplied by a constant.

$$(1/2) \left[- \sum_{i=1}^{n} (y_i - z_i^T c^n)^2 + \sum_{i=1}^{n} (y_i - z_i^T c'^n - c_2 x_i^2)^2 \right] \tag{29}$$

($z_i^T c^n$ and $z_i^T c'^n + c_2 x_i^2$ are merely the approximations to y_i provided by the linear and quadratic functions which are least squares fits to the n data points.) Equations (29) can be rewritten as

$$\sum_{i=1}^{n} y_i [z_i^T c^n - (z_i^T c'^n + c_2 x_i^2)]$$

$$-(1/2) \sum_{i=1}^{n} [(z_i^T c^n)^2 - (z_i^T c'^n + c_2 x_i^2)^2] \quad . \tag{30}$$

This is the usual operation of cross-correlation of the data sequence with a reference function and then the addition of a number not depending explicitly on the data. In a classical simple two hypothesis testing problem, c^n and c'^n are not functions of the data. The reference function is then a deterministic function which is the difference of the mean value functions under the linear and quadratic hypotheses. Also, when dealing with simple hypotheses, the second summation is a constant, i.e., independent of the data. Since we are dealing with composite

hypotheses, the reference function is a function of the data, and the second summation in (30) is also a function of the data. Thus, the decision function in (30) is an estimator-correlator. An interesting simplification of (30) can be made. Through substitution in (30) of (9) and (3), (30) reduces to

$$\sum_{i=1}^{n} y_i[(z_i^T B_n^{-1} \sum_{j=1}^{n} z_j^T c_2 x_j^2) - c_2 x_i^2] \\ -(1/2)[(\sum_{i=1}^{n} z_i^T c_2 x_i^2) B_n^{-1} (\sum_{i=1}^{n} z_i c_2 x_i^2) - \sum_{i=1}^{n} (c_2 x_i^2)^2] \quad . \tag{31}$$

The interpretation of (31) is as follows: $B_n^{-1} \sum_{i=1}^{n} z_i c_2 x_i^2$ is the coefficient vector for a least squares linear fit to the sequence $c_2x_1^2$, $c_2x_2^2$, ..., $c_2x_n^2$. Thus, the line having ith x,y values x_i and $z_i^T B_n^{-1} \sum z_i c_2 x_i^2$ is the line which differs from the quadratic function c_2x^2, at the points $x_1,\ldots,x_n$, least in a sum of error squares sense. Consequently, the decision function reduces to that which would be used for testing the hypothesis that the data is a perturbation of c_2x^2 against the alternative that the data is a perturbation of the line which is most like c_2x^2 at the points x_1, x_2, ..., x_n. The linear hypothesis used here is the most conservative possible! The test is intuitively reasonable since on the one hand, the user is only certain of the c_2x^2 portion of the quadratic hypothesis, and on the other hand, the user knows only that the alternative hypothesis is linear and a worst case hypothesis is therefore a safe one to use. The resulting error rate will of course be higher than if the parameters in the two hypotheses were known. The question that than remains is whether the data is such that the test performs as would a test for hypotheses which were truly simple. The answer is yes, and this is justified in the Appendix.

In summary, then, the stopping test reduces to the test of a simple hypothesis against a simple alternative and the stopping thresholds are set in accordance with the classical theory [14] for such hypotheses.

Case (iii): Case (i) was a stopping rule based on considerable a priori information concerning the problem probability structure. In Case (iii) we function with minimal prior information. The mean value function is assumed to be linear under hypothesis H_o and nonlinear under hypothesis H_1. No additional information is assumed about the mean value function under the alternative hypothesis. A standard approach then would use a chi-square test on the sum of the squares of the errors in the best linear fit to the data. This would involve fixing n and $\alpha > 0$ and deciding H_o or H_1 depending on whether the sum of the squared errors in the best linear fit to n data points lay within or outside a $1 - \alpha$ confidence interval. The drawback of such a test is that if the confidence interval is large and if H_1 is true, the probability of the decision being H_o may be large, i.e., the power of the test may be small. If we wish to design the test to operate with some specified minimum power, then we are back to Case (i). A compromise is to choose n such that the length of the $1 - \alpha$ confidence interval can be fixed at a value we consider to be reasonable for H_o. Then if the confidence interval is not violated, we can be quite certain that H_o is true. A reasonable choice for the length d of the confidence interval might be the following.

Let $\varepsilon(n)$ denote $(\mathcal{E}_n/n)^{1/2}$

(sum of squared errors in linear fit to n data points/n)$^{1/2}$.

That is, $\varepsilon(n)$ is the average error in the linear fit to a data point. A smooth curve of length ℓ might be considered to be a line if it lay within a rectangle of length ℓ and width d with $d/\ell \ll 1$. Since the data for our curves may exhibit considerable fluctuation, we shall consider n data points to have been generated under H_o if

$$\varepsilon(n) < d. \tag{32}$$

Hence, we choose d such that any fairly smooth curve of length roughly ℓ which lies within a rectangle of size d by ℓ is in appearance what we are prepared to call a line. Since the variation in lengths of curves of interest may be considerable, we choose an appropriate number ρ and then determine d from

$$d\rho = \ell \, . \tag{33}$$

Of course, a meaningful d must be greater than the noise standard deviation σ. In summary, then, a confidence interval length d is chosen in terms of specified ρ and ℓ by (33) and n is then determined in order that a $1-\alpha$ confidence interval under H_o have this length. This provides us with a mechanism for being reasonably sure that if our decision is H_o, the associated mean value function is indeed linear, and, furthermore, the probability α of deciding H_1 when H_o is true, is small. We might choose ℓ to be somewhat smaller than the a priori expected lengths of the lines which could be present or simply choose ℓ to be of minimal size yet sufficiently large that n is not excessive. Finally, we mention that these results can be used as a guide for devising a sequential test in an obvious way if the a priori uncertainty in the possible value for ℓ is great.

Case (ii): We point out that prior knowledge of the nature of the mean value function under H_1 which is intermediate between that used in Cases (i) and (iii) can be assumed. Namely, we can assume that the mean value function under H_1 is $c_o + c_1 x + c_2 x^2$ with c_2 unknown and arbitrary. Then the log maximum likelihood ratio test becomes the comparison of

$$\sigma^{-2}[c^{n^T} A_n^T b^n - c'^{n^T} A_n'^T b^n] \tag{34}$$

with 0 where c^n and A_n apply to the quadratic model $z = (1,x,x^2)^T$, and c'^n and A_n' apply to the line model $z' = (1,x)^T$. Since the test statistic is chi-square distributed with 1 degree of freedom, the test can be designed for a specified probability of error when H_o is true. However, if one is willing to assume a quadratic mean value function under H_1, then one may as well treat the problem as Case (i) and we therefore do not investigate Case (ii) further.

Further Comment on Dependence of Goodness of Fit on Number of Data Points

For those applications where the joint likelihood of curve parameters and data is used, it is generally desirable that the functional dependence of this likelihood on the parameter vector be highly peaked in the vicinity of the true underlying parameter vector. Some insight into this dependence on number of data points is provided in this section. The joint likelihood function may be carried along as a performance functional in a sequential pattern recognition problem and it also appears in the simpler problem of recognizing which of two underlying curves of the same degree (or families of such curves) is associated with the data.

We discuss the case of an underlying straight line (the same discussion holds for a higher order curve). Assume the line is parameterized with y as a function of x. The joint density of line parameters and data is that of (25) and (26). The second exponential in (26) is not a function of c. Taking the natural logarithm of (26), we see that

$$(-n/2)\ln(2\pi\sigma^2) - (1/2\sigma^2)\sum_{i=1}^{n}(y_i - \hat{c}_o - \hat{c}_1 x_i)^2$$

is not a function of γ, and has well known behavior since

$\sigma^{-2}\sum_{i=1}^{n}(y_i - \hat{c}_o - \hat{c}_1 x_i)^2$ is chi square distributed with n-2 degrees of freedom. The remainder,

$$\ln[p(H_o, \gamma|w)] - (1/2\sigma^2)\sum_{i=1}^{n}[\hat{c}_o - \gamma_o + (\hat{c}_1 - \gamma_1)x_i)]^2 \qquad (35)$$

is of course a function of γ, and would be the only part of the logarithm of (26) entering into a test as to which of two underlying lines or families of

lines best represented the data. Hence, such a test might be thought of as a comparison of how well the maximum likelihood line fits the lines or families of lines associated with the two hypotheses.

Denote by γ_t the true underlying parameter values. Then

$$-(1/2\sigma^2)\, E\left\{ \sum_{i=1}^{n} [\hat{c}_o - \gamma_o + (\hat{c}_1 - \gamma_1)\, x_i]^2 \;\middle|\; \gamma_t \right\}$$

$$= -(1/2\sigma^2) \sum_{i=1}^{n} E\left\{ [\hat{c}_o - \gamma_{ot} + (\hat{c}_1 - \gamma_{1t})\, x_i]^2 \middle| \gamma_t \right\} - (1/2\sigma^2) \sum_{i=1}^{n} [\gamma_o - \gamma_{ot} + (\gamma_1 - \gamma_{1t}) x_i]^2 \tag{36}$$

The nature of the dependence of (36), the mean of (35), on n shows up on the right of the equality. As a portion of a performance function, (35) becomes useful for those n for which it is small. It can be seen that n should be at least as large as is necessary for the first summation on the right of (36) to be small in magnitude compared with the magnitude of the sum of $\ln[p(H_o,\gamma|w]$ and the second summation on the right of (36). Note that the first summation on the right of (36) is related to the measure of line fit used in the previous section and graphed in Fig. 3. The error measure used there is the first and last terms in the first summation on the right of (36).

Experimental Results on Fitting Ellipses to Noisy Data

Experiments were run using the proposed method for fitting quadratic arcs to noisy data which appeared to be appropriate for approximation by an ellipse.

Two phenomena of interest occurred. First, the optimum quadratic curve was on occasion a hyperbola with each branch fitting a portion of the data. Since no constraint is imposed to restrict the parameters to those for an ellipse, the fitted curve can be an ellipse,

a parabola, or a hyperbola. The parabola essentially never occurs, being a degenerate ellipse or hyperbola. A hyperbola will sometime result in smaller fitting error than an ellipse ... especially if the sampling interval is not uniform along the curve and the data to be fit is somewhat clustered as in Figure 5. Since a single branch of a hyperbola is a satisfactory representation for our purposes, but an approximation involving two branches is not, the problem is to force the best fitting quadratic to be an ellipse or single branch of a hyperbola without giving up the computational advantages of the linear regression methodology. Two approaches to this problem have been tried. The first is to take data points at reasonably equally spaced intervals along the curves being fitted, and then, if necessary, to introduce one or a few artificial data points at appropriate locations to force the desired type of curve without significantly influencing the parameters for the curve of desired type. The approach works in some instances.

In the section on Models, we interpreted the data point error measure when the approximating arc is a portion of an ellipse. A similar interpretation is possible when the approximating curve is a hyperbola. Note that the determinant of the C matrix in that section determines whether the curve (5) = 0 is that for a hyperbola or an ellipse. The curve is an ellipse when the determinant of C is positive and is a hyperbola when the determinant is negative. Suppose C defines a hyperbola and a data point of interest is $v_d = (x_d, y_d)^T$ shown in region I of Figure 6. Then, upon letting v_h denote the point of intersection of the line from v_o to v_d with the hyperbola, we define δ by

$$v_d - v_h = \delta(v_h - v_o) \ . \qquad (37)$$

For two data points each a small perpendicular distance d from the hyperbola, the one furthest from v_o contributes more heavily to the error measure, (4) or the square of (5). If the data point falls in region II in Figure 6, the contribution to (4) is interpreted as follows. If, in the equation defining the hyperbola of Figure 6, the coefficients of x^2 and y^2 are interchanged, the resulting hyperbola, shown in dashed lines in the figure, will be referred to as the complementary hyperbola. Let v_d be the data point shown in region II, and denote by v_c the intersection with the complementary hyperbola of the line from v_o to v_d. Then upon defining δ to be the complex scalar satisfying

$$v_d - i\, v_c = \delta\, i\, v_c, \tag{38}$$

(5) again can be reduced to (6). Note that for (37) and (38), (6) can be rewritten as

$$(x_d/x_h)^2 - 1\ , \quad \text{equivalently,} \quad (y_d/y_h)^2 - 1 \tag{39}$$

and

$$-(x_d/x_c)^2 - 1\ , \quad \text{equivalently,} \quad -(y_d/y_c)^2 - 1\ , \tag{40}$$

respectively. If, for example, the desired fit to data lying about the x axis is an ellipse with major and minor axes more or less parallel to the x and y axes, but the algorithm has fit a hyperbola as in Fig 5 instead, the introduction of a few appropriately placed artificial data points along the line of symmetry lying outside the two branches of the hyperbola (y axis in Fig. 6) will usually then force the best fitting curve to be an ellipse. Good locations for the artificial data points are easily chosen based on (6), for the ellipse, and (39) and (40). The results hold for hyperbolas of abritary orientation. Figure 7 is an example of a single branch hyperbolic fit resulting simply from the use of data points taken at more uniform intervals along the curve

A second and generally successful approach to the problem is the addition of a penalty function to (4) if the least fitting curve consists of portions of two branches of a hyperbola or of an exceedingly small ellipse. For c_1, c_2, c_3 as in (4), the penalty function used was

$$N(c_2^2 - 4\, c_1\, c_3) \tag{41}$$

for $(c_2^2 - 4\, c_1\, c_3) > 0$ and where N is a positive large constant, large compared with the discriminant function $(c_2^2 - 4\, c_1\, c_3)$. With (4) augmented by (41), the equations for the optimal c are still linear. Note that for N large, if $c_2^2 - 4\, c_1\, c_3 > 0$, (41) can impose a large penalty on (4). Hence, the use of this penalty function tends to drive $c_2^2 - 4\, c_1\, c_3$ to 0. Unfortunately, this can happen by forcing all of c_1, c_2, c_3 to be small, and, in fact, this is what happens in practice. That is, an appropriately shaped and sized ellipse seems to be prevented by use of (41), and instead the best fitting curve is forced to be a single branch of a hyperbola with the other branch pushed out toward infinity for large N. This is a satisfactory solution for many purposes. In practice, if (41) is required for a set of n data points, the inversion of B_n at that stage cannot be carried out recursively in terms of B_{n-1}. However, if N is then held fixed, B_{n+1}^{-1}, B_{n+2}^{-1}, etc. can be computed recursively from B_n^{-1}. A reasonable procedure then is to compute two sequences of c^n's, one with a penalty function and fixed N and one without penalty function. When the one without settles down to that for an elliptic curve fit, the sequence based on the penalty function can be discontinued.

The curves in Fig. 8 illustrate the use of (41) in the sequential fit of quadratic curves to data. Data points are found in sequence starting at the upper y-axis and proceeding through the second quadrant down through the third

quadrant. The points were generated as Gaussian perpendicular perturbations of an ellipse. The basic sequential algorithm (9) fits hyperbolas to the first sets of 6 and 7 data points. Both branches are involved in a fit. By use of (41), hyperbolic fits involving only a single branch result. These single branch curves are shown in the figure. The same N was uced in both. The choice of N affects the separation of the two branches of the resulting hyperbola. Furthermore, since single branch hyperbolic fit to the data usually involves that portion of the branch of greatest curvature, N has an affect on the shape of the resulting fit. We see that the fit to six data points is flatter then it really should be thus indicating that a smaller N should have been used which would not have pushed the two branches as far apart and would have permitted the branch used to have greater curvature. Beginning with the ninth data point, the basic algorithm (9) fit an ellipse. Note that beginning with the fit to seven data points, the resulting curves are useful for locating the next data point.

The second phonomenon requiring comment is the following. Suppose data in the vicinity of the origin is best fit by an elliptical arc. Then if the data is translated in the x and y directions by x' and y', respectively, the elliptical arc which best fits the new data is not the translation by (x',y') of the elliptical arc fitting the original data. More specifically the dependence of (6) on ρ, which is a function of v_o and C, leads us to understand that the best fitting ellipse to data removed from the origin will have major and minor axes rotated and scaled differently than would result were the data in the vicinity of the origin. Figure 9 contains overlays of the elliptical arc fits for data at the origin and for the same data translated by (25,25). The solution to the removal of this distortion is to produce a rough elliptical curve fit to the data removed from the vicinity of the origin, then subtract

the coordinates of the center of the rough-fit ellipse from the data points to be used in the fitting process, and then proceed as usual to curve fitting on the translated data. Analogous statements apply if single branch hyperbolic fits are involved.

Conclusions

For the transformed-data linear regression approach to quadratic curve fitting studied in this paper, we interpreted the measure of curve fit error minimized by the procedures. This error is not the sum of squares of the shortest distances between data points and the approximating curve. Nevertheless, the experimental results indicate that the resulting fit is visually satisfactory, our error is small when the standard perpendicular distance error is small, and the measure should be perfectly satisfactory for almost any use.

For straight line fitting, we showed that the data search cost can be an appreciable portion of the total line-fitting cost. The data search cost here was minimal becuase of the tacit assumption of data connectedness so that a data search along a straight line would eventually locate a data point. When such an assumption is no longer true, the data search would have to be over a region rather than along a line and would be more costly.

Though the basic curve-fitting algorithm is fast and requires little computation, our experimental results revealed occasional undesired behavior such as the fitting of two branches of a hyperbola rather than an arc of an ellipse, and distortion in ellipse orientation and shape when the data is noisy and far removed from the origin. Note, neither problem arises if the data contains little noise and the underlying curve is an ellipse. However, when the problem does occur, we found that a satisfactory curve fit involving a single-branch of a hyperbola could be forced through use of a penalty function, and as the curvature

in the data being processed becomes apparent, the original algorithm without penalty function begins to fit an ellipse. The distortion problem was also handled simply as discussed in the section on experimentation.

Curves relating straight-line fitting costs and fitting error were prepared. Asymptotically, error varies as $(\text{cost})^{-1/2}$ as expected. It is assumed here that the length of the data interval being treated is fixed and the option under investigation is the choice of number of data points. The curves shown are parameterized by two parameters: noise standard deviation divided by cell size, σ/Δ, and by noise standard deviation divided by the x interval (or y interval) over which the approximation is made, $\sigma/(x_n-x_1)$. The curve parameter computation costs are derived for straight lines and quadratic curves. It is seen that the latter costs are roughly six times the former. If noisy data search costs are included, the ratio probably will not change much since the line fitting cost may increase by up to 50%, but the quadratic curve fitting cost may almost double due to the peculiarities previously noted. Thus, quadratic curve fitting is relatively expensive. However, for the purpose of recognition of highly variable pictures, curve fitting is often necessary and quadratic curve fitting may be much more useful than piecewise linear fitting followed by polygon recognition and manipulation. Finally, since recognition can often be accomplished using other techniques, e.g., by comparisons, not involving multiplications or divisions, with stored templates, in any recognition cost evaluation an effort must be made to include all significant costs and not just an indication of numbers of multiplications and divisions. Our fitting costs could be reduced by exploiting the fact that the intervals x_i-x_{i-1} used would often be constant and by using truncated variable values and thus perhaps replacing multiplications and divisions by operations requiring less computation time.

The data modeled in this paper consists of thin lines. Thick line drawings can be handled in a number of ways using the methods of this paper or

natural variations. The simplest modification, which should be satisfactory in a variety of applicatiens, would be to replace the data points in a perpendicular slice across a thick line by the center point and then apply the methods of this paper to the transformed data. More efficient methods or more sophisticated methods can be developed depending on how one wishes to model the data generation.

Acknowledgment

David Cooper would like to thank Professor C. D. J. M. Verhagen and his pattern recognition group and Henk Koppelaar of the Technische Hogeschool Delft for many stimulating discussions on the recognition of line drawings; this led to his interest in investigating the relative computation costs of line and curve fitting.

APPENDIX

We show here that the test (31) behaves as if the two hypotheses are simple. The Gaussian distributions under these simple hypotheses have mean vectors $(c_2x_1, c_2x_2, \ldots, c_2x_n)^T$ and $(z_1^T c'^n, z_2^T c'^n, \ldots, z_n^T c'^n)$, respectively, where the latter is a least squares linear fit to the former and

$c'^{n^T} \equiv (c_o'^n, c_1'^n)^T$ is given by $-B_n^{-1} \sum_{i=1}^{n} z^i c_2 x_i^2$. Hence, all the components of the mean value vector for the latter hypothesis are functions of n. Justification for the conclusion that the test performs as though the hypotheses are simple is as follow. We first show that the linear portion of the mean value function of y_i in the test statistic (31) does not contribute to the value of (31). Denote by r^n the n vector having i-th component $z_i^T B_n^{-1} \sum_{j=1}^{n} z_j^T c_2 x_j^2$, and by s^n the n-vector having i-th component $c_2 x_i^2$. Because r^n is a linear least squares fit to s^n, the sum $\sum_{i=1}^{n} (r_i^n - s_i^n)$ is 0 and the inner product of r^n with $r^n - s^n$ is 0. Since the linear portion of the mean value of y^n can be written as a constant vector plus a scalar multiple of r^n, it follows that the inner product of the linear part of the mean value of y^n with $r^n - s^n$ is 0. Hence, (31) would have exactly the same value if y^n had zero mean value vector under H_o and mean value vector $(c_2x_1, \ldots, c_2x_n)^T$ under H_1. Indeed, (31) would have exactly the same value if the mean value of y^n were r^n under H_o and $(c_2x_1, \ldots, c_2x_n)^T$ under H_1. But (31) is the SPRT for these hypotheses, thus concluding the proof.

The authors are with Division of Engineering, Brown University, Providence, RI 02912.

This work was partially funded by the National Aeronautics and Space Administration under grant NSG-5036.

[1] A "T" superscript for a vector denotes the transpose of the vector.

References

[1] H. Koppelaar, "Ontwerp Van Optimalisering Van Letterherkennende Komputer-programma's", Technical Report of the Laboratorium Voor Technische Natuurkunde, Technische Hogeschool Delft, March 1974.

[2] C. H. Chen, "On A Class Of Computationally Efficient Feature Selection Criteria", to appear in the Journal of the Pattern Recognition Society, Vol. 7, No. 1, 1975.

[3] A. Rosenfeld, personal communication concerning his and his students' research results.

[4] T. Pavlidies and S. L. Horowitz, "Segmentation of Plane Curves", IEEE Transactions on Computer, Vol. c-23, No. 8, August 1974; pp. 860-870.

[5] E. M. Riseman and R. W. Ehrich, "Contextual Word Recognition Using Binary Diagrams", IEEE Transactions on Computers, Vol. c-20, No. 4, April 1971; pp. 397-403.

[6] R. Singer, R. Sea, and K. Housewright, "Derivation and Evaluation of Improved Tracking Filters For Use In Dense Multitarget Environments", IEEE Trans. on Information Theory, Vol. IT-20, No. 4, July 1974; pp. 423-432.

[7] M. Yoshida and M. Eden, "Handwritten Chinese Character Recognition By An Analysis-By-Synthesis Method", in Proceedings of the First International Joint Conference on Pattern Recogntion, Washington, DC, October 30 - November 1, 1973; pp. 197-204.

[8] R. Duda and P. Hart, Pattern Classification And Scene Analysis, John Wiley, 1973; pp. 328-334.

[9] W. J. Hemmerle, "Statistical Computations On A Digital Computer, Blaisdel Publishing Co., 1967; p. 75.

[10] A. E. Albert and L. G. Gardner, Stochastic Approximation And Nonlinear Regression, Research Nomograph No. 42, The M.I.T. Press, Cambridge, MA, 1967; pp. 109-112.

[11] IBM System/360 Model 65 Functional Characteristics Manual, Fifth Edition, January 1971, File No. S360-01 GA22-6884-4.

[12] A. F. Mood, Introduction To The Theory Of Statistics, McGraw-Hill, 1950; p. 298.

[13] K. S. Fu, Sequential Methods In Pattern Recognition And Machine Learning, Academic Press, 1968; p. 176.

[14] A. F. Mood, Introduction To The Theory Of Statistics, McGraw-Hill, 1950; pp. 365-382.

[15] W. J. Hemmerle, Statistical Computations On A Digital Computer, Blaisdel Publishing Co., 1967; p. 87.

TABLE 1

Assembly Language Instructions and Average Realization Times for IBM 360/65

Instruction	Mnemonic	Average Approximate Realization Time (μ secs)
Add memory to register	A	2
Add reg. to reg.	AR	1
Branch on index low or equal reg. to mem.	BXLE	2
Compare logical	CLR	1
Divide reg. to reg.	DR	9
Load from mem. to reg.	L	1
Load multiple	LM	1 + R/2*
Load reg. to reg.	LR	1
Multiply reg. to mem.	M	5
Multiply reg. to reg.	MR	4
Subtract mem. to reg.	S	2
Shift left single logical	SLL	1
Subtract reg. to reg.	SR	1
Shift right single logical	SRL	1
Store from reg. to mem.	ST	1
Store multiple	STM	1 + R/2*

* R is the number of registers stored or loaded.

TABLE 2

Numbers of Major Operations and Total CPU Times Per Recursion for Computing c^n, ε^n, $\hat{y}^n$ for Straight Lines and Quadratic Arcs

	Quadratic Arc	Straight Line
c^{n+1}	86AR+5DR+74LR+80M+83SRL + others = 800μ secs.	5AR+2DR+12LR+10MR+11SRL + others = 120μ secs.
ε^{n+1}	4A+5M+others = 41μ secs.	3AR+3MR+3SRL+others = 20μ secs.
$\hat{y}^{n+1}$	3A+DR+5M+5SRL + others = 52μ secs.	2A+M+L+SRL = 10μ secs.
search i pts		8+(i-1)3μ secs.

Programs are written using fixed point arithmetic and it's assumed that all 16 32-bit registers are available.

Figure Captions

Figure 1 Illustration Of The Error Measure In Expression (4).

Figure 2 Illustration Of The Parmeters In Expression (6).

Figure 3 Fitting Cost (cpu time) As A Function Of Error Standard Deviation For Straight Line Fitting To Noisy Data.

Figure 4 Region Of Constant Prior Probability Density For Underlying Lines For Succeeding Data.

Figure 5 Two-Branch Hyperbolic Fit To Noisy Perturbations Of An Elliptic Arc.

Figure 6 Illustration Of The Error Measures In Expressions (39) And (40).

Figure 7 Single Branch Hyperbolic Fit To Noise Perturbations Taken At Roughly Equal Intervals Along An Elliptic Arc.

Figure 8 Sequential Curve Fitting Using A Penalty Function To Force Single Branch Hyperbolic Fits In The Initial Stages.

Figure 9 Overlay Of Elliptic Fit To Data Near The Origin And Elliptic Fit To The Same Data Translated From the Origin.

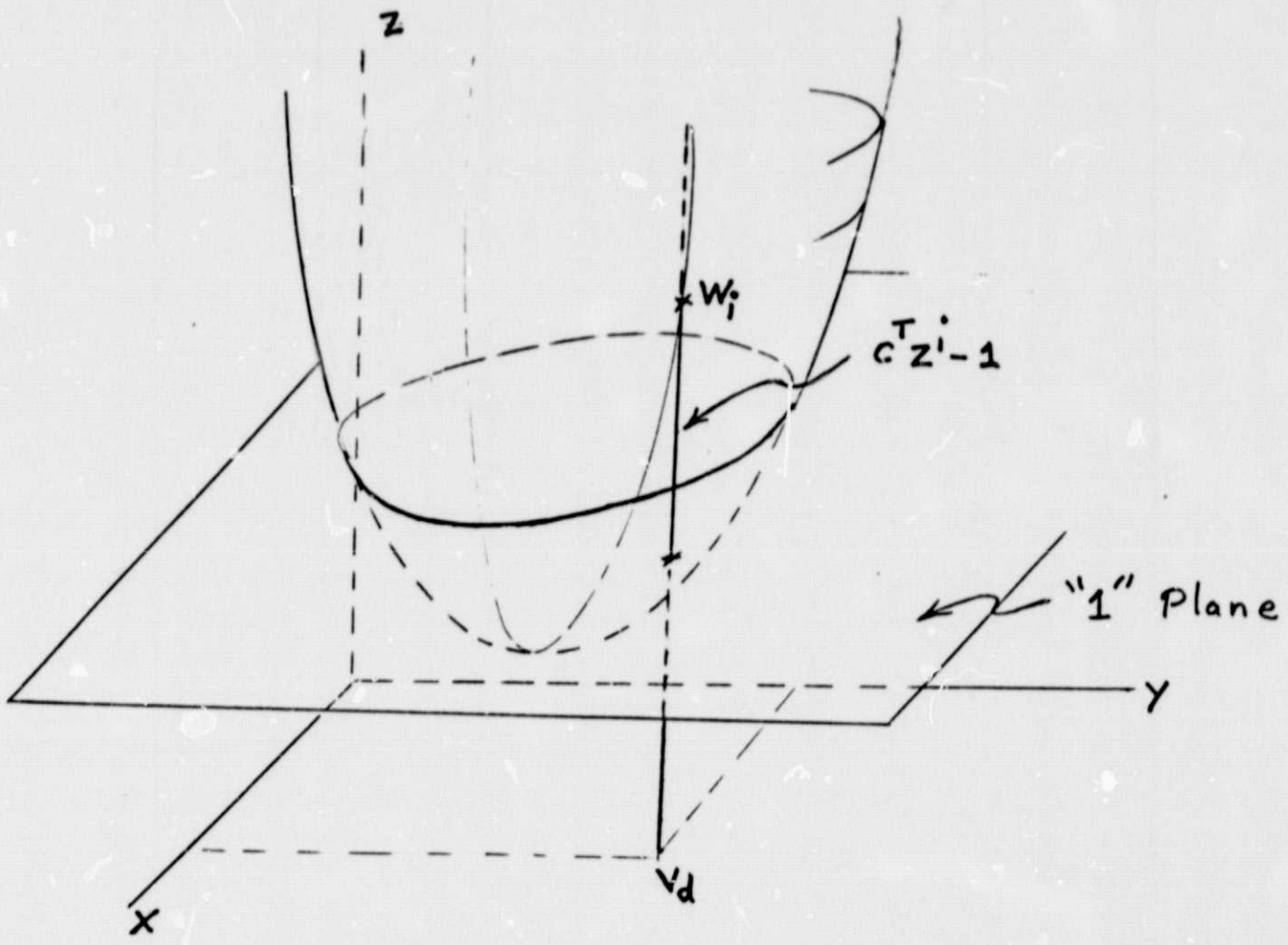

Fig 1

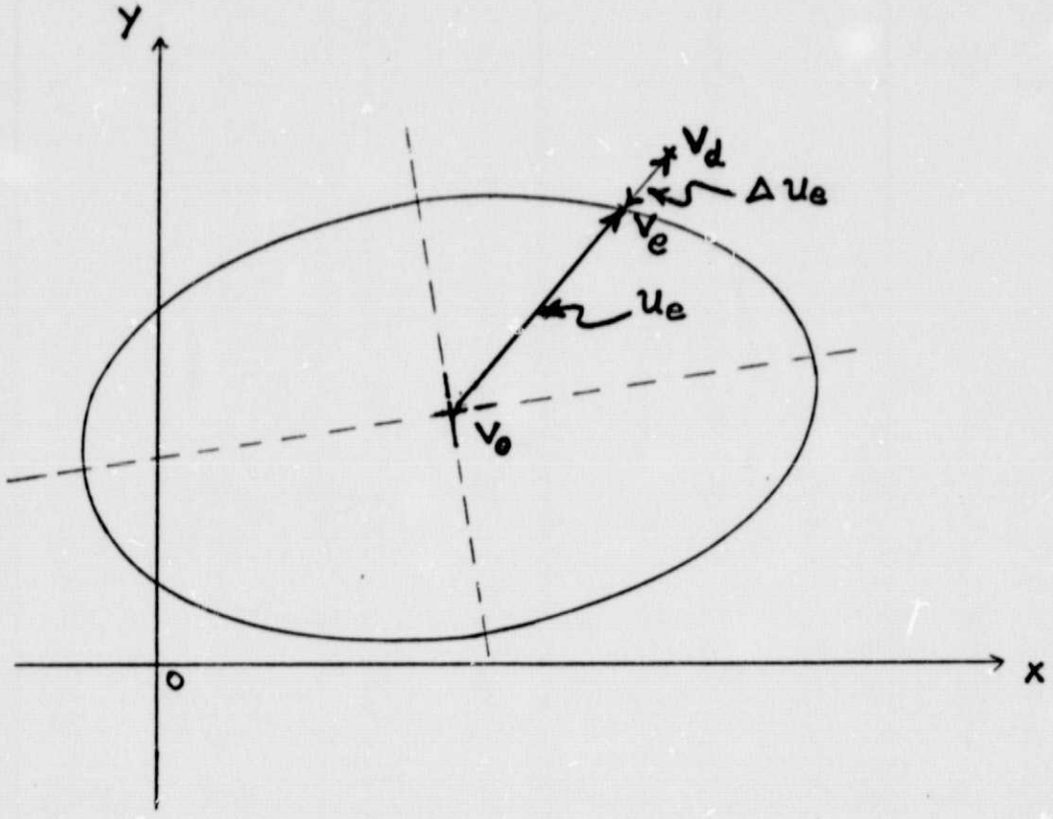

Fig 2

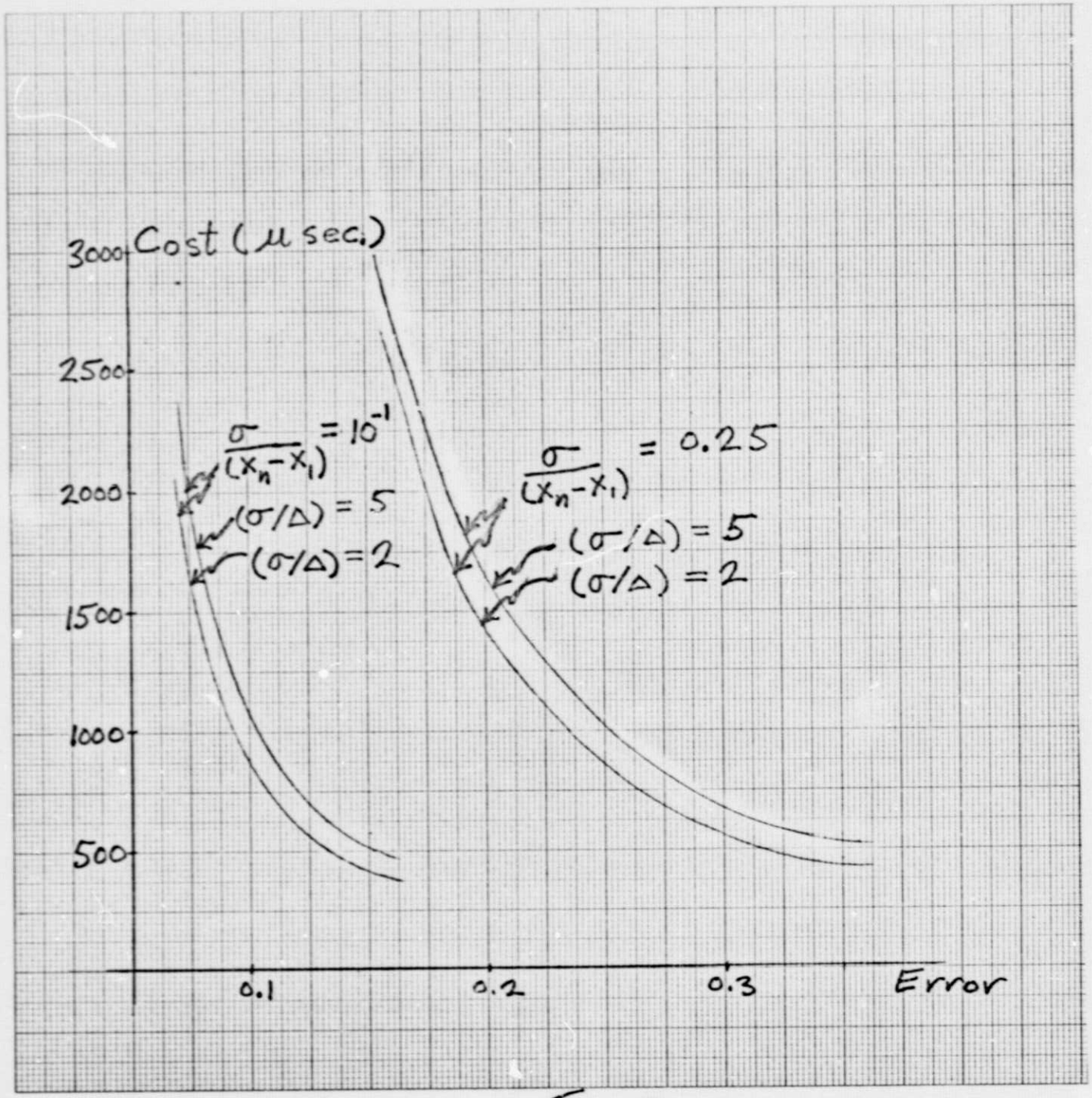

Fig 3

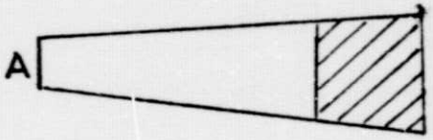

Fig. 4

Fig 5

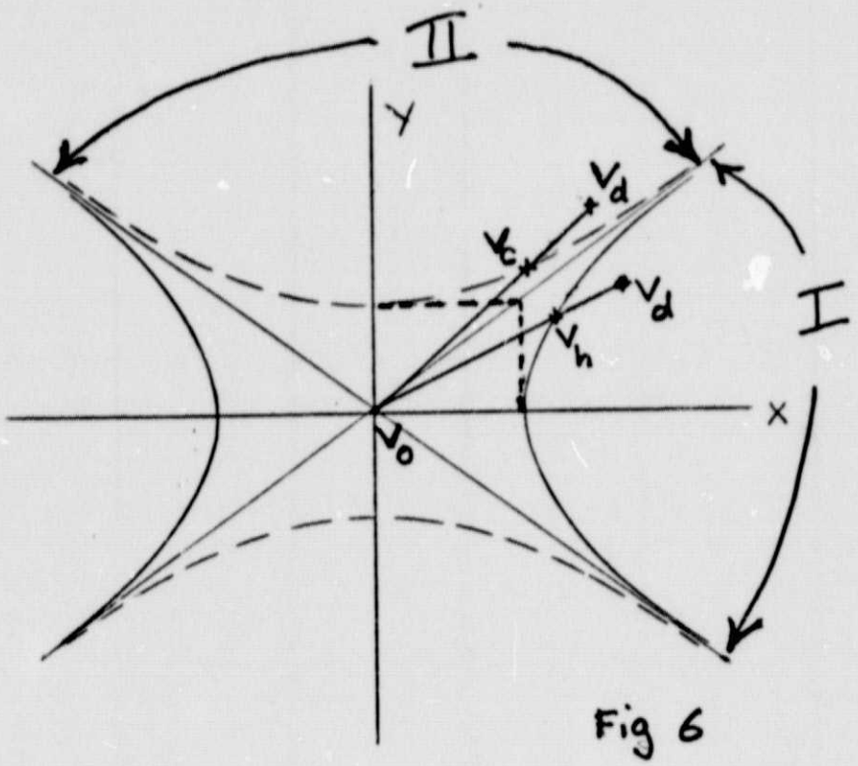

Fig 6

K+E 10 X 10 TO ½ INCH 46 1323
7 X 10 INCHES
MADE IN U.S.A.
KEUFFEL & ESSER CO.
-12
-8
-4
4
8
12

Fig 7

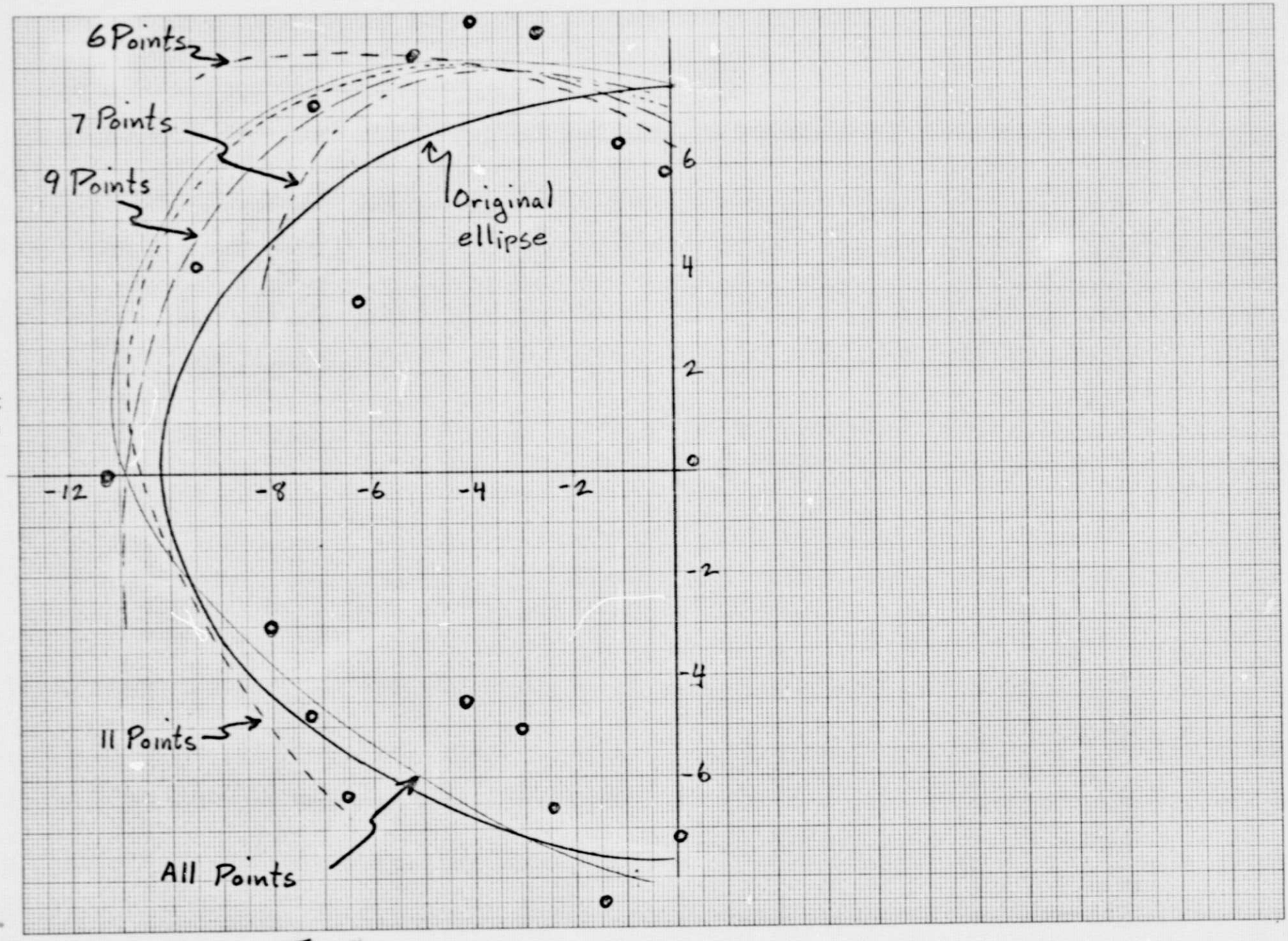

Fig 8

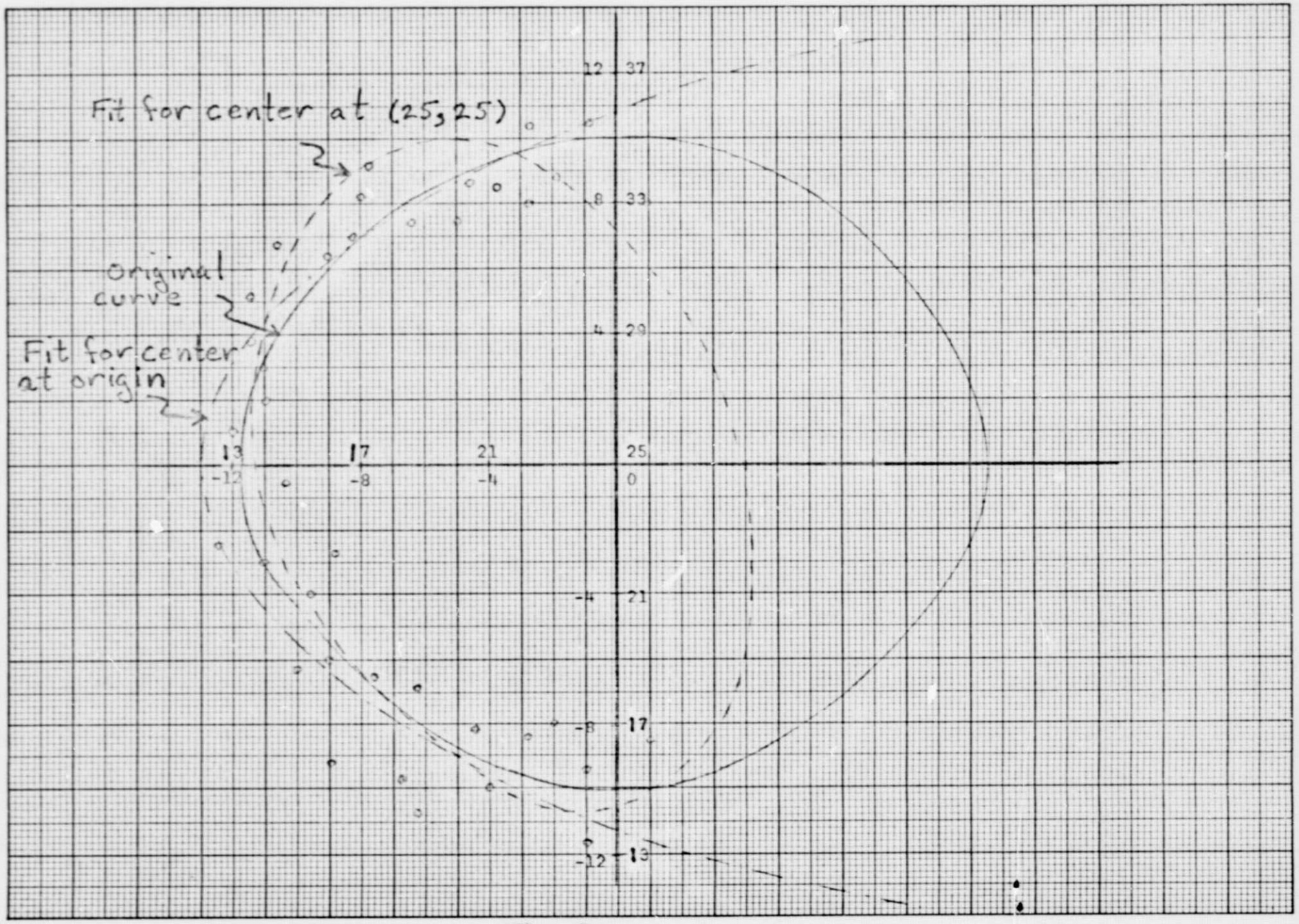

Fig 9